A Local Construction of the Smith Normal Form of a Matrix Polynomial

Jon Wilkening

Department of Mathematics, University of California, Berkeley, CA 94720-3840, USA

Jia Yu

Department of Mathematics, University of California, Berkeley, CA 94720-3840, USA

Abstract

We present an algorithm for computing a Smith form with multipliers of a regular matrix polynomial over a field. This algorithm differs from previous ones in that it computes a local Smith form for each irreducible factor in the determinant separately and then combines them into a global Smith form, whereas other algorithms apply a sequence of unimodular operations to the original matrix row by row (or column by column). The performance of the algorithm in exact arithmetic is reported for several test cases.

Key words: Matrix polynomial, canonical forms, Smith form, Jordan chain, symbolic computation

AMS subject classifications. 68W30, 15A21, 11C20, 11C08

1. Introduction

Canonical forms are a useful tool for classifying matrices, identifying their key properties, and reducing complicated systems of equations to the de-coupled, scalar case. When working with matrix polynomials over a field K, one fundamental canonical form, the Smith form, is defined. It is a diagonalization

$$A(\lambda) = E(\lambda)D(\lambda)F(\lambda) \tag{1}$$

of the given matrix $A(\lambda)$ by unimodular matrices $E(\lambda)$ and $F(\lambda)$ such that the diagonal entries $d_i(\lambda)$ of $D(\lambda)$ are monic polynomials and $d_i(\lambda)$ is divisible by $d_{i-1}(\lambda)$ for $i \geq 2$.

⋆ The authors were supported in part by the Director, Office of Science, Computational and Technology Research, U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Email addresses: wilken@math.berkeley.edu (Jon Wilkening), jiay@math.berkeley.edu (Jia Yu).

This factorization has various applications. The most common one [3] involves solving the system of differential equations

$$A^{(q)}\frac{d^q x}{dt^q} + \cdots + A^{(1)}\frac{dx}{dt} + A^{(0)}x = f(t), \tag{2}$$

where $A^{(0)}$, ..., $A^{(q)}$ are $n \times n$ matrices over $\mathbb{C}$. For brevity, we denote this system by $A(d/dt)x = f$, where $A(\lambda) = A^{(0)} + A^{(1)}\lambda + \cdots + A^{(q)}\lambda^q$. Assume for simplicity that $A(\lambda)$ is regular, i.e. $\det(A(\lambda))$ is not identically zero, and that (1) is a Smith form of $A(\lambda)$. The system (2) is then equivalent to

$$\begin{pmatrix} d_1(\frac{d}{dt}) & & \\ & \ddots & \\ & & d_n(\frac{d}{dt}) \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix},$$

where $y = F(d/dt)x(t)$ and $g = E^{-1}(d/dt)f(t)$. Note that $E^{-1}(\lambda)$ is a matrix polynomial over $\mathbb{C}$ due to the unimodularity of $E(\lambda)$. This system splits into n independent scalar ordinary differential equations

$$d_i\Big(\frac{d}{dt}\Big)y_i(t) = g_i(t), \qquad 1 \le i \le n,$$

and the solution of (2) is then given by $x = F^{-1}(d/dt)y$, where $F^{-1}(\lambda)$ is also a matrix polynomial over $\mathbb{C}$.

Smith forms of linear matrix polynomials are also related to the concept of similarity of matrices. A fundamental theorem in matrix theory states that two square matrices A and B over a field K are similar if and only if their characteristic matrix polynomials $\lambda I - A$ and $\lambda I - B$ have the same Smith form $D(\lambda)$. [1, 3].

Other applications of this canonical form include finding the Frobenius form of a matrix A over a field by computing the invariant factors of the linear matrix polynomial $\lambda I - A$ [13, 15].

The computation of Smith forms of matrices over $\mathbb{Q}[\lambda]$ is a widely studied topic. Kannan [8] gave a method for computing the Smith form with repeated triangularizations of the matrix polynomial over $\mathbb{Q}$. Kaltofen, Krishnamoorthy and Saunders [6] gave the first polynomial time algorithm for the Smith form (without multipliers) using the Chinese remainder theorem. A new class of probabilistic algorithms (the Monte Carlo algorithms) were proposed by Kaltofen, Krishnamoorthy and Saunders [6, 7]. They showed that by pre-multiplying the given matrix polynomial by a randomly generated constant matrix on the right, the Smith form with multipliers is obtained with high probability by two steps of computation of the Hermite form. A Las Vegas algorithm given by Storjohann and Labahn [10, 11] significantly improved the complexity by rapidly checking the correctness of the result of the KKS algorithm. Villard [12, 14] established the first deterministic polynomial-time method to obtain the Smith form with multipliers by explicitly computing a good-conditioning matrix that replaces the random constant matrix in the Las Vegas algorithm. He also applied the method in Marlin, Labhalla and Lombardi [9] to obtain useful complexity bounds for the algorithm.

We propose a new deterministic algorithm for the computation of Smith forms of matrix polynomials over a field in Section 3. Our approach differs from previous methods in that we begin by constructing local diagonal forms that we later combine to obtain

a (global) post-multiplier. Since we do not discuss complexity bounds, we compare the performance of our algorithm to Villard's method with good conditioning in Section 4, and discuss the reasons for the increase in speed. In Appendix A, we present a parallel algebra theory that connects this work to [16], and give a variant of the algorithm in which all operations are done in the field K rather than manipulating polynomials as such.

2. Preliminaries

In this section, we describe the theory of Smith forms of matrix polynomials over a field K, which follows the definition in [3] over $\mathbb{C}$. We also give a brief review of the theory of Jordan chains as well as Bézout's identity, which play an important role in our algorithm for computing Smith forms of matrix polynomials.

2.1. Smith Forms

Suppose $A(\lambda) = \sum_{k=0}^{q} A^{(k)}\lambda^k$ is an $n \times n$ matrix polynomial, where $A^{(k)}$ are $n \times n$ matrices whose entries are in a field K. Assuming that $A(\lambda)$ is *regular*, i.e. the determinant of $A(\lambda)$ is not identically zero, the following theorem is proved in [3] (for $K = \mathbb{C}$).

Theorem 1. *There exist matrix polynomials $E(\lambda)$ and $F(\lambda)$ over K of size $n \times n$, with constant nonzero determinants, such that*

$$A(\lambda) = E(\lambda)D(\lambda)F(\lambda), \qquad D(\lambda) = \mathrm{diag}[d_1(\lambda), \ldots, d_n(\lambda)], \tag{3}$$

where $D(\lambda)$ is a diagonal matrix with monic scalar polynomials $d_i(\lambda)$ over K such that $d_i(\lambda)$ is divisible by $d_{i-1}(\lambda)$.

Since $E(\lambda)$ and $F(\lambda)$ have constant nonzero determinants, (3) is equivalent to

$$U(\lambda)A(\lambda)V(\lambda) = D(\lambda), \tag{4}$$

where $U(\lambda) := (E(\lambda))^{-1}$ and $V := (F(\lambda))^{-1}$ are also matrix polynomials over K.

Definition 2. The representation in (3) or (4), or often $D(\lambda)$ alone, is called a *Smith form* of $A(\lambda)$. Square matrix polynomials with constant nonzero determinants like $E(\lambda)$ and $F(\lambda)$ are called *unimodular*.

The diagonal matrix $D(\lambda)$ in the Smith form is unique, while the representation (3) is not. Suppose that

$$\Delta(\lambda) := \det(A(\lambda)) \tag{5}$$

can be decomposed into prime elements $p_1(\lambda), \ldots, p_l(\lambda)$ in the principal ideal domain $K[\lambda]$, that is, $\Delta(\lambda) = c\prod_{j=1}^{l} p_j(\lambda)^{\kappa_j}$ with κ_j a positive integer and $c \in K \setminus \{0\}$. Then the $d_i(\lambda)$ are given by

$$d_i(\lambda) = \prod_{j=1}^{l} p_j(\lambda)^{\kappa_{ji}}, \qquad (1 \le i \le n)$$

for some integers $0 \le \kappa_{j1} \le \cdots \le \kappa_{jn}$ satisfying $\sum_{i=1}^{n} \kappa_{ji} = \kappa_j$ for $j = 1, \ldots, l$.

We now define a *local Smith form* for $A(\lambda)$ at $p(\lambda)$. Let $p(\lambda) = p_j(\lambda)$ be one of the irreducible factors of $\Delta(\lambda)$ and define $\alpha_i = \kappa_{ji}$, $\mu = \kappa_j$. Generalizing the case that $p(\lambda) = \lambda - \lambda_j$, we call μ the algebraic multiplicity of $p(\lambda)$.

Theorem 3. *Suppose A is an $n \times n$ matrix over $K[\lambda]$ and $p(\lambda)$ is an irreducible factor of $\Delta(\lambda)$. There exist $n \times n$ matrix polynomials $E(\lambda)$ and $F(\lambda)$ such that*

$$A(\lambda) = E(\lambda)D(\lambda)F(\lambda), \qquad D(\lambda) = \begin{pmatrix} p(\lambda)^{\alpha_1} & & 0 \\ & \ddots & \\ 0 & & p(\lambda)^{\alpha_n} \end{pmatrix}, \tag{6}$$

where $0 \le \alpha_1 \le \cdots \le \alpha_n$ are nonnegative integers and $p(\lambda)$ does not divide $\det[E(\lambda)]$ *or* $\det[F(\lambda)]$.

$E(\lambda)$ and $F(\lambda)$ are not uniquely determined in a local Smith form. In particular, we can always choose $F(\lambda)$ to be unimodular by absorbing the missing parts of $D(\lambda)$ in Theorem 1 into $E(\lambda)$. Then the local Smith form of $A(\lambda)$ at $p(\lambda)$ is given by

$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \tag{7}$$

where $V(\lambda) := F(\lambda)^{-1}$ is a matrix polynomial.

2.2. Jordan Chains

Finding a local Smith form of a matrix polynomial over $\mathbb{C}$ at $p(\lambda) = \lambda - \lambda_0$ is equivalent to finding a canonical system of Jordan chains [2, 16] for $A(\lambda)$ at λ_0. We now generalize the notion of Jordan chain to the case of an irreducible polynomial over a field K.

Definition 4. Suppose $A(\lambda)$ is an $n \times n$ matrix polynomial over a field K and $p(\lambda)$ is irreducible in $K[\lambda]$. A vector polynomial $x(\lambda) \in K[\lambda]$ of the form

$$x(\lambda) = x^{(0)}(\lambda) + p(\lambda)x^{(1)}(\lambda) + \cdots + p(\lambda)^{\alpha-1}x^{(\alpha-1)}(\lambda) \tag{8}$$

with $\deg x^{(k)}(\lambda) < s := \deg p(\lambda)$ and $\alpha \ge 1$, is called a *Jordan chain of length* α for $A(\lambda)$ at $p(\lambda)$ if

$$A(\lambda)x(\lambda) = O(p(\lambda)^{\alpha}) \tag{9}$$

and $x^{(0)}(\lambda) \not\equiv 0$. The meaning of (9) is that each component of $A(\lambda)x(\lambda)$ is divisible by $p(\lambda)^{\alpha}$. Any vector polynomial $x(\lambda)$ satisfying (9) such that $p(\lambda) \nmid x(\lambda)$ is called a *root function of order* α for $A(\lambda)$ at $p(\lambda)$. We generally truncate or zero-pad $x(\lambda)$ to have α terms in an expansion in powers of $p(\lambda)$ when referring to it as a Jordan chain. If K can be embedded in $\mathbb{C}$, (9) implies that over $\mathbb{C}$, $x(\lambda)$ is a root function of $A(\lambda)$ of order α at each root λ_j of $p(\lambda)$ simultaneously.

Definition 5. Several vector polynomials $\{x_j(\lambda)\}_{j=1}^{\nu}$ form a *system of root functions* at $p(\lambda)$ if

$$\begin{aligned} &A(\lambda)x_j(\lambda) = O(p(\lambda)^{\alpha_j}), \qquad (\alpha_j \ge 1, \quad 1 \le j \le \nu) \\ &\text{the set } \{\dot{x}_j(\lambda)\}_{j=1}^{\nu} \text{ is linearly independent in } M/pM \text{ over } R/pR, \\ &\qquad\qquad \text{where } R = K[\lambda],\ M = R^n,\ \dot{x}_j = x_j + pM. \end{aligned} \tag{10}$$

It is called *canonical* if (1) $\nu = \dim \ker \dot{A}$, where $\dot{A}$ is the linear operator on M/pM induced by A; (2) $x_1(\lambda)$ is a root function of maximal order α_1; and (3) for $i > 1$, $x_i(\lambda)$ has maximal order α_i among all root functions $x(\lambda) \in M$ such that $\dot{x}$ is linearly independent of $\dot{x}_1, \ldots, \dot{x}_{i-1}$ in M/pM. The integers $\alpha_1 \ge \cdots \ge \alpha_\nu$ are uniquely determined by $A(\lambda)$.

Definition 6. An *extended system of root functions* $x_1(\lambda),\ldots,x_n(\lambda)$ is a collection of vector polynomials satisfying (10) with ν replaced by n and α_j allowed to be zero. The extended system is said to be canonical if, as before, the orders α_j are chosen to be maximal among root functions not in the span of previous root functions in M/pM; the resulting sequence of numbers $\alpha_1 \geq \cdots \geq \alpha_\nu \geq \alpha_{\nu+1} = \cdots = \alpha_n = 0$ is uniquely determined by $A(\lambda)$.

Given such a system (not necessarily canonical), we define the matrices

$$V(\lambda) = [x_1(\lambda), \ldots, x_n(\lambda)], \tag{11}$$
$$D(\lambda) = \operatorname{diag}[p(\lambda)^{\alpha_1}, \ldots, p(\lambda)^{\alpha_n}], \tag{12}$$
$$E(\lambda) = A(\lambda)V(\lambda)D(\lambda)^{-1}. \tag{13}$$

All singularities of $E(\lambda)$ are clearly removable. The following theorem shows that aside from a reversal of the convention for ordering the α_j, finding a local Smith form is equivalent to finding an extended canonical system of root functions:

Theorem 7. *The following three conditions are equivalent:*
(1) the columns $x_j(\lambda)$ of $V(\lambda)$ form an extended canonical system of root functions for $A(\lambda)$ at $p(\lambda)$ (up to a permutation of columns).
(2) $p(\lambda) \nmid \det[E(\lambda)]$.
(3) $\sum_{j=1}^{n} \alpha_j = \mu$, where μ is the algebraic multiplicity of $p(\lambda)$ in $\Delta(\lambda)$.

This theorem is proved e.g. in [2] for the case that $K = \mathbb{C}$. The proof over a general field K is identical, except that the following lemma is used in place of invertibility of $E(\lambda_0)$. This lemma also plays a fundamental role in our construction of Jordan chains and local Smith forms.

Lemma 8. *Suppose K is a field, p is an irreducible polynomial in $R = K[\lambda]$, and $E = [y_1, \ldots, y_n]$ is an $n \times n$ matrix with columns $y_j \in M = R^n$. Then $p \nmid \det(E) \Leftrightarrow \{\dot{y}_1, \ldots, \dot{y}_n\}$ are linearly independent in M/pM over R/pR.*

Proof. The $\dot{y}_j$ are linearly independent iff the determinant of $\dot{E}$ (considered as an $n \times n$ matrix with entries in the field R/pR) is non-zero. But

$$\det \dot{E} = \det E + pR, \tag{14}$$

where $\det E$ is computed over R. The result follows. □

2.3. Bézout's Identity

As $K[\lambda]$ is a principal ideal domain, Bézout's Identity holds, which is our main tool for combining local Smith forms into a single global Smith form. We define the notation $\gcd(f_1, \ldots, f_l)$ to be 0 if each f_j is zero, and the monic greatest common divisor (GCD) of $f_1, \ldots, f_l$ over $K[\lambda]$, otherwise.

Theorem 9. (Bézout's Identity) *For any two polynomials f_1 and f_2 in $K[\lambda]$, where K is a field, there exist polynomials g_1 and g_2 in $K[\lambda]$ such that*

$$g_1 f_1 + g_2 f_2 = \gcd(f_1, f_2). \tag{15}$$

Bézout's Identity can be extended to combinations of more than two polynomials:

Theorem 10. (Generalized Bézout's Identity) *For any scalar polynomials $f_1, \dots, f_l$ in $K[\lambda]$, there exist polynomials $g_1, \dots, g_l$ in $K[\lambda]$ such that*

$$\sum_{j=1}^{l} g_j f_j = \gcd(f_1, \dots, f_l).$$

The polynomials g_j are called extended greatest common divisors *of $\{f_1, \dots, f_l\}$.*

In particular, suppose we have l distinct prime elements $\{p_1, \dots, p_l\}$ in $K[\lambda]$, and f_j is given by $f_j = \prod_{k \neq j}^{l} p_k^{\beta_k}$, where $\beta_1, \dots, \beta_l$ are given positive integers and the notation $\prod_{k \neq j}^{l}$ indicates a product over all indices $k = 1, \dots, l$ except $k = j$. Then $\gcd(f_1, \dots, f_l) = 1$, and we can find $g_1, \dots, g_l$ over $K[\lambda]$ such that

$$\sum_{j=1}^{l} g_j f_j = 1. \tag{16}$$

In this case, the extended greatest common divisors g_j are uniquely determined by requiring $\deg(g_j) < s_j \beta_j$, where $s_j = \deg(p_j)$. The formula (16) modulo p_k shows that g_k is not divisible by p_k.

3. An Algorithm for Computing a (global) Smith Form

In this section, we describe an algorithm for computing a Smith form of a regular $n \times n$ matrix polynomial $A(\lambda)$ over a field K. We have in mind the case where $K = \mathbb{C}$, $\mathbb{R}$, $\mathbb{Q}$ or $\mathbb{Q} + i\mathbb{Q} \subset \mathbb{C}$, but the construction works for any field. The basic procedure follows several steps, which will be explained further below:

- Step 0. Compute $\Delta(\lambda) = \det A(\lambda)$ and decompose it into irreducible factors

$$\Delta(\lambda) = \text{const} \cdot p_1(\lambda)^{\kappa_1} \dots p_l(\lambda)^{\kappa_l}. \tag{17}$$

- Step 1. Compute a local Smith form

$$A(\lambda) V_j(\lambda) = E_j(\lambda) \begin{pmatrix} p_j(\lambda)^{\kappa_{j1}} & & 0 \\ & \ddots & \\ 0 & & p_j(\lambda)^{\kappa_{jn}} \end{pmatrix} \tag{18}$$

  for each factor $p_j(\lambda)$ of $\Delta(\lambda)$.
- Step 2. Find a linear combination $B_n(\lambda) = \sum_{j=1}^{l} g_j(\lambda) f_j(\lambda) V_j(\lambda)$ using extended GCDs of $f_j(\lambda) = \prod_{k \neq j}^{l} p_j(\lambda)^{\kappa_{kn}}$ so that the columns of $B_n(\lambda)$ form an extended canonical system of root functions for $A(\lambda)$ with respect to each $p_j(\lambda)$.
- Step 3. Eliminate extraneous zeros from $\det\big[A(\lambda) B_n(\lambda)\big]$ by finding a unimodular matrix $V(\lambda)$ such that $B_1(\lambda) = V(\lambda)^{-1} B_n(\lambda)$ is lower triangular. We will show that $A(\lambda)V(\lambda)$ is then of the form $E(\lambda)D(\lambda)$ with $E(\lambda)$ unimodular and $D(\lambda)$ as in (3).
  Note that the diagonal entries in the matrix polynomial $D(\lambda)$ are given by

$$d_i(\lambda) = \prod_{j=1}^{l} p_j(\lambda)^{\kappa_{ji}}, \quad i = 1, \dots, n$$

once we know the local Smith forms. This allows us to order the columns once and for all in Step 2.

3.1. A Local Smith Form Algorithm (Step 1)

In this section, we show how to generalize the construction in [16] (for finding a canonical system of Jordan chains for an analytic matrix function $A(\lambda)$ over $\mathbb{C}$ at $\lambda_0 = 0$) to finding a local Smith form for a matrix polynomial $A(\lambda)$ with respect to an irreducible factor $p(\lambda)$ of $\Delta(\lambda) = \det[A(\lambda)]$. The new algorithm reduces to the "exact arithmetic" version of the previous algorithm when $p(\lambda) = \lambda$. In Appendix A, we present a variant of the algorithm that is easier to implement than the current approach, and is closer in spirit to the construction in [16], but is less efficient by a factor of $s = \deg p$.

Our goal is to find matrices $V(\lambda)$ and $E(\lambda)$ such that $p(\lambda)$ does not divide $\det[V(\lambda)]$ or $\det[E(\lambda)]$, and such that

$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \qquad D(\lambda) = \operatorname{diag}[p(\lambda)^{\alpha_1}, \ldots, p(\lambda)^{\alpha_n}], \tag{19}$$

where $0 \le \alpha_1 \le \cdots \le \alpha_n$. In our construction, $V(\lambda)$ will be unimodular, which reduces the work in Step 3 of the high level algorithm, the step in which extraneous zeros are removed from the determinant of the combined local Smith forms.

We start with $V(\lambda) = I_{n\times n}$ and perform a sequence of column operations on $V(\lambda)$ that preserve its determinant (up to a sign) and systematically increase the orders α_i in $D(\lambda)$ in (19) until $\det[E(\lambda)]$ no longer contains a factor of $p(\lambda)$. This can be considered a "breadth first" construction of a canonical system of Jordan chains, in contrast to the "depth first" procedure described in Definition 5.

The basic algorithm is presented in Figure 1. Here $R = K[\lambda]$ is a principal ideal domain and $M = R^n$ is a free R-module of rank n:

$$R = K[\lambda], \qquad M = R^n. \tag{20}$$

Since p is irreducible, R/pR is a field and M/pM is a vector space over this field. We adopt C notation for integer arithmetic and use / and % to denote the quotient and remainder of polynomials:

$$g = f/p, \quad r = f\,\%\,p \qquad \Leftrightarrow \qquad f = gp + r, \quad \deg r < \deg p. \tag{21}$$

The idea of the algorithm is to run through the columns of V in turn and "accept" columns whenever the leading term of the residual $A(\lambda)x_i(\lambda)$ is linearly independent of its predecessors; otherwise we find a linear combination of previously accepted columns to cancel this leading term and cyclically rotate the column to the end for further processing. Note that for each k, we cycle through each unaccepted column exactly once: after rotating a column to the end, it will not become active again until k has increased by one. At the start of the *while* loop, we have the invariants

(1) Ax_m is divisible by p^k, $\quad (i \le m \le n)$.
(2) $Ax_m = p^{\alpha_m} y_m + O(p^{\alpha_m+1})$, $\quad (1 \le m < i)$.
(3) if $i \ge 2$ then $\{\dot{y}_m\}_{m=1}^{i-1}$ is linearly independent in M/pM over R/pR.

The third property is guaranteed by the *if* statement, and the second property follows from the first due to the definition of α_i and y_i in the algorithm. The first property is

Algorithm 1. (Local smith form, preliminary version)

$k = 0,\ \ i = 1,\ \ V = [x_1, \ldots, x_n] = I_{n\times n}$
while $i \le n$
  $r_{k-1} = n + 1 - i$ $r_{k-1} :=$ *dim. of space of J. chains of length* $\ge k$
  for $j = 1, \ldots, r_{k-1}$
    $y_i = (Ax_i/p^k)\%\, p$ *define* y_i *so* $Ax_i = p^k y_i + O(p^{k+1})$
    if the set $\{\dot y_1, \ldots, \dot y_i\}$ is linearly independent in M/pM over R/pR
      $\alpha_i = k, \quad i = i + 1$ *accept* x_i *and* y_i*, define* α_i
    else
      find $\dot a_1, \ldots, \dot a_{i-1} \in R/pR$ so that $\dot y_i - \sum_{m=1}^{i-1} \dot a_m \dot y_m = \dot 0$
    ⋆ $x_i^{(new)} = x_i^{(old)} - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m x_m$
      $\mathrm{tmp} = x_i, \quad x_m = x_{m+1},\ (m = i, \ldots, n-1), \quad x_n = \mathrm{tmp}$
    end if
  end for j
  $k = k + 1$
end while
$\beta = k - 1, \quad r_\beta = 0$ $\beta := \alpha_n =$ *maximal Jordan chain length*

Fig. 1. Algorithm for computing a local Smith form.

obviously true when $k = 0$; it continues to hold each time k is incremented due to step ⋆, after which $Ax_i^{(new)}$ is divisible by p^{k+1}:

$$Ax_i^{(old)} - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m Ax_m = p^k y_i + O(p^{k+1}) - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m \Big(p^{\alpha_m} y_m + O(p^{\alpha_m+1})\Big)$$
$$= p^k \Big(y_i - \sum_{m=1}^{i-1} a_m y_m\Big) + O(p^{k+1}) = O(p^{k+1}).$$

This equation is independent of which polynomials $a_m \in R$ are chosen to represent $\dot a_m \in R/pR$, but different choices will lead to different (equally valid) Smith forms; in practice, we choose the unique representatives such that $\deg a_m < s$, where

$$s = \deg p. \tag{22}$$

This choice of the a_m leads to two additional invariants of the *while* loop, namely
(4) $\deg x_m \le \max(sk - 1, 0), \qquad (i \le m \le n)$,
(5) $\deg x_m \le \max(s\alpha_m - 1, 0), \quad (1 \le m < i)$,
which are easily proved inductively by noting that

$$\deg(p^{k-\alpha_m} a_m x_m) \le s(k - \alpha_m) + (s - 1) + \deg(x_m). \tag{23}$$

The *while* loop eventually terminates, for at the end of each loop (after k has been incremented) we have produced a unimodular matrix $V(\lambda)$ such that

$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \qquad D = \mathrm{diag}[p^{\alpha_1}, \ldots, p^{\alpha_{i-1}}, \underbrace{p^k, \ldots, p^k}_{r_{k-1}\ \text{times}}]. \tag{24}$$

Hence, the algorithm must terminate before k exceeds the algebraic multiplicity μ of $p(\lambda)$ in $\Delta(\lambda)$:

$$k \le \left(\sum_{m=1}^{i=1} \alpha_i\right) + (n+1-i)k \le \mu, \qquad \Delta(\lambda) = f(\lambda)p(\lambda)^\mu, \quad p \nmid f. \tag{25}$$

In fact, we can avoid the last iteration of the *while* loop if we change the test to

$$\textbf{while } \left[\left(\sum_{m=1}^{i-1} \alpha_i\right) + (n+1-i)k\right] < \mu$$

and change the last line to

$$\beta = k, \qquad \alpha_m = k, \quad (i \le m \le n), \qquad r_{\beta-1} = n+1-i, \qquad r_\beta = 0.$$

We know the remaining columns of V will be accepted without having to compute the remaining y_i or check them for linear independence. When the algorithm terminates, we will have found a unimodular matrix $V(\lambda)$ satisfying (49) such that the columns of

$$\dot{E}(\lambda) = [\dot{y}_1(\lambda), \ldots, \dot{y}_n(\lambda)]$$

are linearly independent in M/pM over R/pR. By Lemma 8, $p(\lambda) \nmid \det[E(\lambda)]$, as required.

To implement the algorithm, we must find an efficient way to compute y_i, test for linear independence in M/pM, find the coefficients a_m to cancel the leading term of the residual, and update x_i. Motivated by the construction in [16], we interpret the loop over j in Algorithm 1 as a single nullspace calculation.

Let us define $R_l = \{a \in R \ : \ \deg a < l\}$ and $M_l = R_l^n$, both viewed as vector spaces over K. Then we have an isomorphism Λ of vector spaces over K

$$\begin{aligned} \Lambda &: (M_s)^k \to M_{sk}, \\ \Lambda(x^{(0)}; \ldots; x^{(k-1)}) &= x^{(0)} + px^{(1)} + \cdots + p^{k-1}x^{(k-1)}. \end{aligned} \tag{26}$$

At times it will be convenient to identify R_{ls} with R/p^lR and M_{ls} with M/p^lM to obtain ring and module structures for these spaces. We also expand

$$A = A^{(0)} + pA^{(1)} + \cdots + p^qA^{(q)}, \tag{27}$$

where $A^{(j)}$ is an $n \times n$ matrix with entries in R_s. By properties (4) and (5) of the *while* loop, we may write $x_i = \Lambda(x_i^{(0)}; \ldots; x_i^{(\alpha)})$ with $\alpha = \max(k-1, 0)$. Since Ax_i is divisible by p^k in Algorithm 1, we have

$$y_i = (Ax_i/p^k) \% p = \sum_{j=0}^{k} \left[A^{(k-j)}x_i^{(j)}\right] \% p + \sum_{j=0}^{k-1} \left[A^{(k-1-j)}x_i^{(j)}\right]/p. \tag{28}$$

The matrix-vector multiplications $A^{(k-j)}x_i^{(j)}$ are done in the ring R (leading to vector polynomials of degree $\le 2s-2$) before the quotient and remainder are taken. When $k = 0$, the second sum should be omitted, and when $k \ge 1$, the $j = k$ term in the first sum can be dropped since $x_i^{(k)} = 0$ in the algorithm. In practice, we test all the active columns $\dot{y}_i, \ldots, \dot{y}_n \in M/pM$ for linear independence of their predecessors *simultaneously* as follows. If $k = 0$ we have

$$[y_1, \ldots, y_n] = A^{(0)}. \tag{29}$$

Otherwise $k \ge 1$ and we have computed the matrix X_{k-1} with columns $\left(x_m^{(0)}; \ldots; x_m^{(k-1)}\right)$ for $i \le m \le n$ such that $\Lambda(X_{k-1})$ (acting column by column) represents the last r_{k-1} columns of $V(\lambda)$ at the start of the *while* loop in Algorithm 1. Then by (28),

$$[y_i, \ldots, y_n] = [A^{(k)}, \ldots, A^{(1)}]X_{k-1} \% p + [A^{(k-1)}, \ldots, A^{(0)}]X_{k-1}/p. \tag{30}$$

As before, the matrix multiplications are done in the ring R before the quotient and remainder are computed to obtain the components of y_m, which belong to R_s. To test for linear independence, we find the kernel of the matrix $\dot{\mathcal{A}}_k$, where

$$\mathcal{A}_k = \begin{cases} A^{(0)}, & k = 0, \\ \big[\mathcal{A}_{k-1}\ ,\ [y_i, \ldots, y_n]\big], & 1 \le k \le n. \end{cases} \tag{31}$$

To compute this kernel, we find the reduced row-echelon form of $\dot{\mathcal{A}}_k$ using Gauss-Jordan elimination over the field R/pR. Multiplication and division in R/pR are easily carried out using the companion matrix of p. If we define

$$\gamma : K^s \to R/pR, \qquad \gamma\big(x^{(0)}; \ldots; x^{(s-1)}\big) = x^{(0)} + \cdots + \lambda^{s-1} x^{(s-1)} + pR, \tag{32}$$

we can pull back the field structure of R/pR to K^s to obtain

$$\begin{aligned} xy &= \gamma(x)(S)y = [x^{(0)} I + x^{(1)} S + \cdots + x^{(s-1)} S^{s-1}]y = [y, Sy, \ldots, S^{s-1}y]x, \\ x/y &= [y, Sy, \ldots, S^{s-1}y]^{-1}x, \qquad\qquad x = \big(x^{(0)}; \ldots; x^{(s-1)}\big) \in K^s, \end{aligned} \tag{33}$$

where

$$S = \begin{pmatrix} 0 & \ldots & 0 & -a_0 \\ 1 & \ddots & \vdots & \vdots \\ & \ddots & 0 & -a_{s-2} \\ 0 & & 1 & -a_{s-1} \end{pmatrix}, \quad p(\lambda) = a_0 + a_1\lambda + \cdots + a_{s-1}\lambda^{s-1} + \lambda^s \tag{34}$$

is the companion matrix of p, and represents multiplication by λ in R/pR. The matrix $[y, Sy, \ldots, S^{s-1}y]$ is invertible when $y \neq 0$ since a non-trivial vector x in its kernel would lead to non-zero polynomials $\gamma(x), \gamma(y) \in R/pR$ whose product is zero (mod p), which is impossible as p is irreducible.

The reduced row-echelon form of $\dot{\mathcal{A}}_k$ can be interpreted as a tableau telling which columns of $\dot{\mathcal{A}}_k$ are linearly independent of their predecessors (the accepted columns), and also giving the linear combination of previously accepted columns that will annihilate a linearly dependent column. On the first iteration (with $k = 0$), step $\star$ in Algorithm 1 will build up the matrix

$$X_0 = \mathrm{null}(\dot{\mathcal{A}}_0), \tag{35}$$

where null$(\cdot)$ is the standard algorithm for computing a basis for the nullspace of a matrix from the reduced row-echelon form (followed by a truncation to replace the elements in R/pR of this nullspace matrix with their representatives in R_s). But rather than rotating these columns to the end as in Algorithm 1, we now *append* the corresponding y_i to the end of $\mathcal{A}_{k-1}$ to form $\mathcal{A}_k$ for $k \ge 1$. The "dead" columns left behind (not accepted, not active) serve only as placeholders, causing the resulting matrices $\mathcal{A}_k$ to be nested. The leading columns of rref$(\dot{\mathcal{A}}_k)$ will then coincide with rref$(\dot{\mathcal{A}}_{k-1})$, and the nullspace matrices will also be nested:

$$\begin{pmatrix} X_0 & V_1 & \cdots & V_{k-1} & V_k \\ 0 & [U_1; 0] & \cdots & [U_{k-1}; 0] & U_k \end{pmatrix} := \mathrm{null}(\dot{\mathcal{A}}_k). \tag{36}$$

Note that $\mathcal{A}_k$ is $n \times (n + R_{k-1})$, where

$$R_{-1} = 0, \qquad R_k = r_0 + \cdots + r_k = \dim\ker \dot{\mathcal{A}}_k, \qquad (k \ge 0). \tag{37}$$

We also see that X_0 is $n \times r_0$, V_k is $n \times r_k$, and U_k is $r_{k-1} \times r_k$. Since the dimension of the kernel cannot increase by more than the number of columns added,

$$r_k \le r_{k-1}, \qquad (k \ge 0). \tag{38}$$

If column i of $\dot{\mathcal{A}}_k$ is linearly dependent on its predecessors, the coefficients a_m used in step $\star$ of Algorithm 1 are precisely the (truncations of the) coefficients that appear in column i of $\mathrm{rref}(\dot{\mathcal{A}}_k)$. The corresponding null vector (i.e. column of $[V_k; U_k]$) contains the negatives of these coefficients in the rows corresponding to the previously accepted columns of $\dot{\mathcal{A}}_k$, followed by a 1 in row i; see Figure 2. Thus, in step $\star$, if $k \ge 1$ and we write $x_m = \Lambda\big(x_m^{(0)}; \ldots; x_m^{(\alpha)}\big)$ with $\alpha = \max(\alpha_m - 1, 0)$, the update

$$x_i^{(new)} = x_i^{(old)} - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m x_m, \qquad a_m x_m \,\%\, p^{\alpha_m+1} = \Lambda\big(z^{(0)}; \ldots; z^{(\alpha_m)}\big),$$

$$z^{(j)} = \begin{cases} \big(a_m x_m^{(0)}\big) \,\%\, p, & j = 0, \\ \big(a_m x_m^{(j)}\big) \,\%\, p + \big(a_m x_m^{(j-1)}\big)/p, & 1 \le j < \alpha_m, \\ \big(a_m x_m^{(j-1)}\big)/p, & j = \alpha_m \text{ and } \alpha_m > 0, \end{cases}$$

is equivalent to

$$\begin{aligned} X_k = \left\{ \left[\iota^k(X_{-1})\,,\, \iota^{k-1}\rho(X_0)\,,\, \ldots,\, \iota^0\rho(X_{k-1})\right] \begin{pmatrix} V_k \\ U_k \end{pmatrix} \right\} \% \, p \\ + \Big(\left[\iota^k(X_0)\,,\, \ldots\,,\, \iota^1(X_{k-1})\right] U_k \Big)/p, \end{aligned} \tag{39}$$

where $\iota, \rho : (M_s)^l \to (M_s)^{l+1}$ act column by column, padding them with zeros:

$$\iota(x) = (0; x), \qquad \rho(x) = (x; 0), \qquad x \in (M_s)^l, \quad 0 \in M_s. \tag{40}$$

Here $\Lambda\iota\Lambda^{-1}$ is multiplication by p, which embeds $M_{ls} \cong M/p^l M$ in $M_{(l+1)s} \cong M/p^{l+1}M$ as a module over R, while ρ is an embedding of vector spaces over K (but not an R-module morphism). If we define the matrices $\mathbb{X}_0 = X_0$ and

$$\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k] = \left[\begin{pmatrix} 0_{nk \times r_0} \\ X_0 \end{pmatrix}, \begin{pmatrix} 0_{n(k-1) \times r_1} \\ X_1 \end{pmatrix}, \ldots, \begin{pmatrix} X_k \end{pmatrix} \right], \quad (k \ge 1), \tag{41}$$

then (39) simply becomes

$$X_k = [(\mathbb{X}_{k-1} U_k) \,\%\, p; V_k] + \iota(\mathbb{X}_{k-1} U_k)/p. \tag{42}$$

As in (30) above, the matrix multiplications are done in the ring R before the quotient and remainder are computed to obtain X_k. Finally, we line up the columns of X_{k-1} with the last r_{k-1} columns of $\dot{\mathcal{A}}_k$ and extract (i.e. accept) columns of X_{k-1} that correspond to new, linearly independent columns of $\dot{\mathcal{A}}_k$. We denote the matrix of extracted columns by $\widetilde{X}_{k-1}$. At the completion of the algorithm, the unimodular matrix $V(\lambda)$ that puts $A(\lambda)$ in local Smith form is given by

$$V(\lambda) = \big[\Lambda(\widetilde{X}_{-1}), \ldots, \Lambda(\widetilde{X}_{\beta-1})\big]. \tag{43}$$

The final algorithm is presented in Figure 3. In the steps marked $\bullet$, we can avoid recomputing the reduced row-echelon form of the first $n + R_{k-2}$ columns of $\dot{\mathcal{A}}_k$ by storing

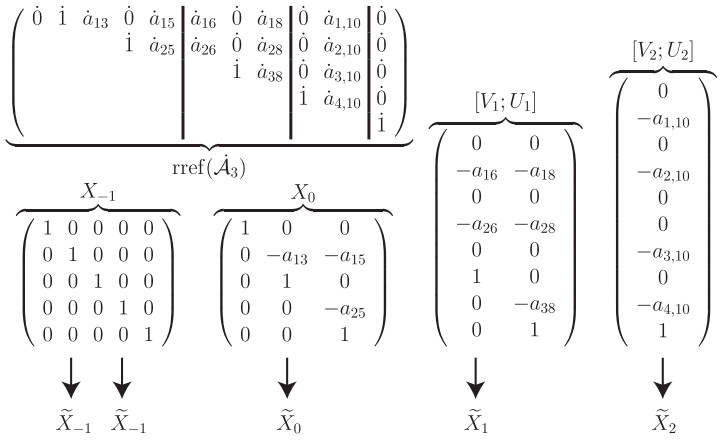

Fig. 2. The reduced row-echelon form of $\dot{\mathcal{A}}_\beta$ contains all the information necessary to construct $V(\lambda) = [\Lambda(\widetilde{X}_{-1}), \ldots, \Lambda(\widetilde{X}_{s-1})]$. An arrow from a column $[v; u]$ of $[V_k; U_k]$ indicates that the vector $\big([(\mathbb{X}_{k-1}u)\,\%\,p; v] + \iota(\mathbb{X}_{k-1}u)/p\big)$ should be added to $\widetilde{X}_k$.

Algorithm 2. (Local smith form, final version)

$k = 0$
$\mathcal{A}_0 = A^{(0)}$
$X_0 = \mathbb{X}_0 = \text{null}(\dot{\mathcal{A}}_0)$
$r_0 = R_0 = \text{num\_cols}(\mathbb{X}_0)$ (number of columns)
$\widetilde{X}_{-1} = [e_{j_1}, \ldots, e_{j_{n-r_0}}]$, (columns j_i of rref($\dot{\mathcal{A}}_0$) start new rows)
while $R_k < \mu$ (μ = algebraic multiplicity of p)
$k = k + 1$
- $\mathcal{A}_k = \big(\, \mathcal{A}_{k-1}\ ,\ \big[A^{(k)}, \ldots, A^{(1)}\big] X_{k-1}\,\%\,p + \big[A^{(k-1)}, \ldots, A^{(0)}\big] X_{k-1}/p \,\big)$
- $[V_k; U_k] =$ new columns of null($\dot{\mathcal{A}}_k$) beyond those of null($\dot{\mathcal{A}}_{k-1}$)

$r_k = \text{num\_cols}(U_k)$, ($U_k$ is $R_{k-1} \times r_k$)
$R_k = R_{k-1} + r_k$
$X_k = [(\mathbb{X}_{k-1}U_k)\,\%\,p; V_k] + \iota(\mathbb{X}_{k-1}U_k)/p$ (X_k is $n(k+1) \times r_k$)
$\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k]$ ($\mathbb{X}_k$ is $n(k+1) \times R_k$)
$\widetilde{X}_{k-1} = X_{k-1}(:, [j_1, \ldots, j_{r_{k-1}-r_k}])$, (columns $n + R_{k-2} + j_i$ of rref($\dot{\mathcal{A}}_k$) start new rows)
end while
$\beta = k + 1$ (maximal Jordan chain length)
$\widetilde{X}_{\beta-1} = X_{\beta-1}$
$V(\lambda) = \big[\Lambda(\widetilde{X}_{-1}), \ldots, \Lambda(\widetilde{X}_{\beta-1})\big]$

Fig. 3. Algorithm for computing a unimodular local Smith form.

the sequence of Gauss-Jordan transformations [4] that reduced $\dot{\mathcal{A}}_{k-1}$ to row-echelon form. To compute $[V_k; U_k]$, we need only apply these transformations to the new columns of $\dot{\mathcal{A}}_k$ and then proceed with the row-reduction algorithm on these final columns. Also, if A_0 is large and sparse, rather than reducing to row-echelon form, one could find kernels using

an LU factorization designed to handle singular matrices. This would allow the use of graph theory (clique analysis) to choose pivots in the Gaussian elimination procedure to minimize fill-in. We also note that if $\Delta(\lambda)$ contains only one irreducible factor, the local Smith form is a (global) Smith form of $A(\lambda)$; steps 2 and 3 can be skipped in that case.

3.2. Algorithms for the Extended GCD Problem

The extended Euclidean algorithm is widely applied to solve the extended GCD problem for two polynomials in $K[\lambda]$, e.g. gcdex in Maple. The algorithm requires $O(P(d)\log d)$ arithmetic operations in K to solve the problem, where f_1 and f_2 both have degrees no greater than d and $P(d)$ is the number of field operations in K required to multiply two polynomials of degree $d-1$ in $K[\lambda]$. Using standard polynomial multiplication, we have $P(d) = d^2$, while a fast algorithm [10] uses $P(d) = d(\log d)(\log\log d)$.

The extended GCD problem (16) for more than two polynomials can be solved by applying the extended Euclidean algorithm to all the polynomials simultaneously; see below. A variant of this approach is to focus on the two lowest degree polynomials until one is reduced to zero; we then repeat until all but one is zero. In practice, we use the function 'hermite' in Maple.

Suppose we have n polynomials $f_1, \ldots, f_n$ in $K[\lambda]$. We find $f_i \neq 0$ with the lowest degree. Each f_j with $j \neq i$ can then be written as

$$f_j - q_j f_i = r_j,$$

where q_j is the quotient of f_j divided by f_i, and r_j is the remainder. Hence, we have

$$\begin{pmatrix} 1 & & -q_1 & & \\ & \ddots & \vdots & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ & & -q_n & & 1 \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_i \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ f_i \\ \vdots \\ r_n \end{pmatrix}.$$

Denote the matrix by Q_1. We repeat this procedure on $(r_1; \ldots; f_i; \ldots; r_n)$ until there is only one nonzero entry in the vector, and we obtain

$$Q_k Q_{k-1} \ldots Q_1 \begin{pmatrix} f_1 \\ \\ \vdots \\ \\ f_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ r \\ \vdots \\ 0 \end{pmatrix}. \tag{44}$$

The row of $Q_k Q_{k-1} \ldots Q_1$ with the same index as r divided by the leading coefficient of r gives a solution of the extended GCD problem.

3.3. From Local to Global (Step 2)

Now that we have a local Smith form (18) for every irreducible factor $p_j(\lambda)$ of $\Delta(\lambda)$, we can apply the algorithm in Section 3.2 to obtain a family of polynomials $\{g_j(\lambda)\}_{j=1}^{l}$ with $\deg(g_j(\lambda)) < s_j\kappa_{jn}$, where $s_j = \deg(p_j)$, such that

$$\sum_{j=1}^{l}\left[g_j(\lambda)\prod_{k=1,k\neq j}^{l}p_k(\lambda)^{\kappa_{kn}}\right] = 1, \tag{45}$$

where $p_j(\lambda)^{\kappa_{jn}}$ is the last entry in the diagonal matrix of the local Smith form at $p_j(\lambda)$. The integers κ_{jn} are positive. We define a matrix polynomial $B_n(\lambda)$ via

$$B_n(\lambda) = \sum_{j=1}^{l}\left[g_j(\lambda)V_j(\lambda)\prod_{k\neq j}^{l}p_k(\lambda)^{\kappa_{kn}}\right]. \tag{46}$$

The main result of this section is stated as follows.

Proposition 11. *The matrix polynomial $B_n(\lambda)$ defined by* (46) *has two properties:*

(1) Let $b_{ni}(\lambda)$ be the ith column vector polynomial of $B_n(\lambda)$. Then $A(\lambda)b_{ni}(\lambda)$ is divisible by $d_i(\lambda)$, where $d_i(\lambda) = \prod_{j=1}^{l}p_j(\lambda)^{\kappa_{ji}}$ is the ith diagonal entry in $D(\lambda)$ of the Smith form.

(2) $\det[B_n(\lambda)]$ is not divisible by $p_j(\lambda)$ for $j = 1,\ldots,l$.

Proof. 1. Let $v_{ji}(\lambda)$ be the ith column of $V_j(\lambda)$. Then $A(\lambda)v_{ji}(\lambda)$ is divisible by $p_j(\lambda)^{\kappa_{ji}}$ and

$$b_{ni}(\lambda) = \sum_{j=1}^{l}\left[\prod_{k\neq j}^{l}p_k(\lambda)^{\kappa_{kn}}\right]g_j(\lambda)v_{ji}(\lambda).$$

Since $\kappa_{jn} \geq \kappa_{ji}$ for $1 \leq i \leq n$ and $1 \leq j \leq l$, $A(\lambda)b_{ni}(\lambda)$ is divisible by $d_i(\lambda)$.

2. The local Smith form construction ensures that $p_j(\lambda) \nmid \det[V_j(\lambda)]$ for each $1 \leq j \leq l$. Equation (45) modulo $p_j(\lambda)$ shows that $p_j(\lambda) \nmid g_j(\lambda)$. By definition,

$$\begin{aligned}\det[B_n(\lambda)] &= \det\left(\left[b_{n1}(\lambda)\,,\,\ldots\,,\,b_{nn}(\lambda)\right]\right) = \det\left(\left[b_{ni}(\lambda)\right]_{i=1}^{n}\right)\\ &= \det\left(\left[\sum_{j'=1}^{l}\left(\prod_{k\neq j'}^{l}p_k(\lambda)^{\kappa_{kn}}\right)g_{j'}(\lambda)v_{j'i}(\lambda)\right]_{i=1}^{n}\right).\end{aligned}$$

Each term in the sum is divisible by $p_j(\lambda)$ except $j' = j$. Thus, by multi-linearity,

$$\det[B_n(\lambda)]\,\%\,p_j(\lambda) = \left(\left[\prod_{k\neq j}^{l}p_k(\lambda)^{\kappa_{kn}}\right]^{n}\left[g_j(\lambda)\right]^{n}\det\left[V_j(\lambda)\right]\right)\%\,p_j(\lambda) \neq 0,$$

as claimed. □

Remark 12. It is possible for $\det[B_n(\lambda)]$ to be non-constant; however, its irreducible factors will be distinct from $p_1(\lambda),\ldots,p_l(\lambda)$.

Remark 13. Rather than building $B_n(\lambda)$ as a linear combination (46), we may form $B_n(\lambda)$ with columns

$$b_{ni}(\lambda) = \sum_{j=1}^{l}\left[\prod_{k\neq j}^{l}p_k(\lambda)^{\max(\kappa_{ki},1)}\right]g_{ij}(\lambda)v_{ji}(\lambda), \qquad (1 \leq i \leq n),$$

where $\{g_{ij}\}_{j=1}^{l}$ solves the extended GCD problem

$$\sum_{j=1}^{l}\left[g_{ij}(\lambda)\prod_{k\neq j}^{l}p_k(\lambda)^{\max(\kappa_{ki},1)}\right]=1.$$

The two properties proved above also hold for this definition of $B_n(\lambda)$. This modification can significantly reduce the size of the coefficients in the computation when there is a wide range of Jordan chain lengths. But if κ_{ji} only changes slightly for $1\leq i\leq n$, this change will not significantly affect the running time as solving for the local Smith forms is the most expensive step in our method.

3.4. Construction of Unimodular Matrix Polynomials (Step 3)

Given a vector polynomial $[f_1(\lambda);\ldots;f_n(\lambda)]\in K[\lambda]^n$, we can use the extended GCD algorithm to find a unimodular matrix $Q(\lambda)$ such that $Q(\lambda)f(\lambda)=[0;\ldots;0;r(\lambda)]$, where $r=\gcd(f_1,\ldots,f_n)$. Explicitly, we apply one additional transformation Q_{k+1} to (44) to swap row i with row n, where i is the index of r in (44), and define $Q=Q_{k+1}Q_k\ldots Q_1$. We apply this procedure to the last column of $B_n(\lambda)$ and define $V_n(\lambda)=Q(\lambda)^{-1}$. The resulting matrix

$$B_{n-1}(\lambda):=V_n(\lambda)^{-1}B_n(\lambda)$$

is zero above the main diagonal in column n. We then apply this procedure to the first $n-1$ components of column $n-1$ of $B_{n-1}(\lambda)$ to get a new $Q(\lambda)$, and define

$$V_{n-1}(\lambda)=\left(\begin{array}{ccc:c}&&&0\\&Q(\lambda)^{-1}&&\vdots\\&&&0\\\hdashline 0&\cdots&0&1\end{array}\right). \tag{47}$$

It follows that $B_{n-2}(\lambda):=V_{n-1}(\lambda)^{-1}B_{n-1}(\lambda)$ is zero above the main diagonal in columns $n-1$ and n. Continuing in this fashion, we obtain unimodular matrices $V_n(\lambda),\ldots,V_2(\lambda)$ such that

$$A(\lambda)B_n(\lambda)=A(\lambda)\underbrace{V_n(\lambda)\cdots V_2(\lambda)}_{V(\lambda)}V_2(\lambda)^{-1}\cdots\underbrace{V_n(\lambda)^{-1}B_n(\lambda)}_{B_{n-1}(\lambda)}=A(\lambda)V(\lambda)B_1(\lambda),$$

where $V(\lambda)$ is unimodular, $B_1(\lambda)$ is lower triangular, and

$$\det[B_1(\lambda)]=\pm\det[B_n(\lambda)]. \tag{48}$$

The matrix $V(\lambda)$ puts $A(\lambda)$ in Smith form:

Proposition 14. *There is a unimodular matrix polynomial $E(\lambda)$ such that*

$$A(\lambda)V(\lambda)=E(\lambda)D(\lambda), \tag{49}$$

where $D(\lambda)$ is of the form (3).

Proof. Let r_{mi} denote the entry of $B_1(\lambda)$ in the mth row and ith column. Define $y_i(\lambda)$ and $z_i(\lambda)$ to be the ith columns of $A(\lambda)V(\lambda)$ and $A(\lambda)V(\lambda)B_1(\lambda)$, respectively, so that

$$z_i(\lambda)=y_i(\lambda)r_{ii}(\lambda)+\sum_{m=i+1}^{n}y_m(\lambda)r_{mi}(\lambda),\qquad(1\leq i\leq n). \tag{50}$$

By Proposition 11, $z_i(\lambda)$ is divisible by $d_i(\lambda)$ for $1 \le i \le n$ and $p_j(\lambda) \nmid \det[B_1(\lambda)]$ for $1 \le j \le l$. It follows that the diagonal entries $r_{ii}(\lambda)$ of $B_1(\lambda)$ are relatively prime to each of the $d_i(\lambda)$. As $d_n(\lambda)$ divides $y_n(\lambda)r_{nn}(\lambda)$ and is relatively prime to $r_{nn}(\lambda)$, it divides $y_n(\lambda)$ alone. Now suppose $1 \le i < n$ and we have shown that $d_m(\lambda)$ divides $y_m(\lambda)$ for $i < m \le n$. Then since $d_i(\lambda)$ divides $d_m(\lambda)$ for $m > i$ and $r_{ii}(\lambda)$ is relatively prime to $d_i(\lambda)$, we conclude from (50) that $d_i(\lambda)$ divides $y_i(\lambda)$. By induction, $d_i(\lambda)$ divides $y_i(\lambda)$ for $1 \le i \le n$. Thus, there is a matrix polynomial $E(\lambda)$ such that (49) holds. Because $V(\lambda)$ is unimodular and $\det[A(\lambda)] = \text{const} \cdot \det[D(\lambda)]$, it follows that $E(\lambda)$ is also unimodular, as claimed. □

4. Performance Comparison

In this section ,we compare our algorithm to Villard's method with good conditioning [14], which is another deterministic sequential method for computing Smith forms with multipliers, and to 'smith' in Maple. All the algorithms are implemented in exact arithmetic using Matlab's symbolic toolbox using the variant of Algorithm 2 given in Appendix A.

To evaluate the performance of these methods, we generate several groups of diagonal matrices $D(\lambda)$ over $\mathbb{Q}$ and multiply them on each side by unimodular matrices of the form $L(\lambda)U(\lambda)P$, where P is a permutation matrix, L is unit lower triangular, and U is unit upper triangular, both with off diagonal entries of the form $\lambda - i$ with $i \in \{-10, \ldots, 10\}$ a random integer. Each process is repeated five times for each $D(\lambda)$ and the median running time is recorded.

We use several parameters in the comparison, including the size n of the square matrix $A(\lambda)$, the bound d of the degrees of the entries in $A(\lambda)$, the number l of irreducible factors in $\det(A(\lambda))$, and the largest order κ_{jn}. The computation of local Smith forms turns out to be the most expensive part of our method. Therefore, the running time mostly depends on n, l and κ_{jn}. The cost of Villard's method with good condition is a function of n and d, as well as the size of the coefficients, which we do not discuss in this paper.

Our first group of test matrices $D_n(\lambda)$ are of the form

$$D_n(\lambda) = \text{diag}[1, \ldots, 1, \lambda, \lambda(\lambda-1), \lambda^2(\lambda-1), \lambda^2(\lambda-1)^2],$$

where n increases starting from 4. Hence, we have $d = 8$, $l = 2$, and $\kappa_{1n} = \kappa_{2n} = 2$ all fixed. (The unimodular matrices in the construction of A each have degree 2). A comparison of the various algorithms is plotted in Figure 4. Since we apply the algorithm using the symbolic toolbox in Matlab, the time of computing the leading terms of the expansion $A(\lambda) = A^{(0)} + A^{(1)}p_j + A^{(2)}p_j^2 + \cdots$ is far larger than it should be (due apparently to storage of arbitrary precision numbers as strings rather than maple objects). In Figure 4, we plot both the actual running time ($\times$) and a corrected running time ($+$) in which the timer is turned off for the computation of the coefficients $A^{(j)}$. The slopes of the lines give the exponent α in $t(n) = Cn^\alpha$. We did not implement the re-use strategy for computing the reduced row-echelon form of $\mathcal{A}_k$ by storing the Gauss-Jordan transformations used to obtain $\text{rref}(\mathcal{A}_{k-1})$, and then continuing with only the new columns of $\mathcal{A}_k$. This is because the built-in function rref is much faster than can be achieved by a user defined matlab code for the same purpose; therefore, our running times could be reduced by a factor of about 2 in this test if we had access to the internal rref code.

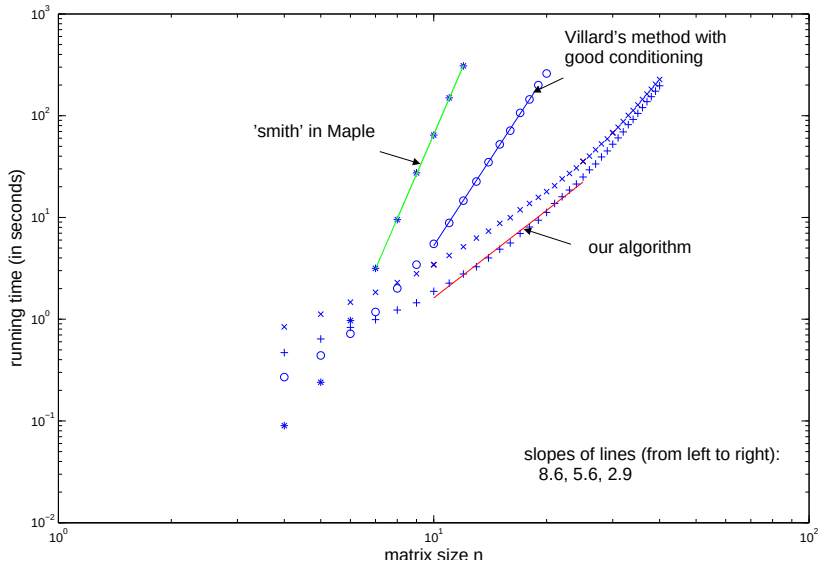

Fig. 4. Running time vs. matrix size n.

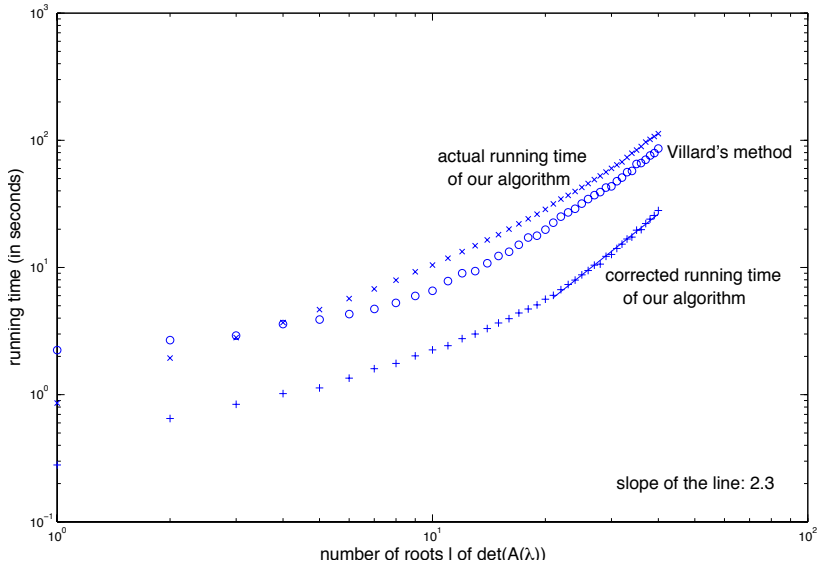

Fig. 5. Running time vs. number of roots l of $\det(A(\lambda))$.

For the second test, we use test matrices $D_k(\lambda)$ of size 9×9, where

$$D_k(\lambda) = \mathrm{diag}[1, \ldots, 1, \textstyle\prod_{j=1}^{k}(\lambda - j)], \qquad (k = 1, 2, \ldots).$$

In other words, $n = 9$, $l = k$, $d = k + 4$ and $\kappa_{jn} = 1$ for $1 \le j \le k$. We omit '`smith`' in Maple for this case, which is much slower than the other two methods. The running time of Villard's method does not directly rely on l but increases as d grows. Also, note that the actual running time of our algorithm is slightly slower than Villard's method,

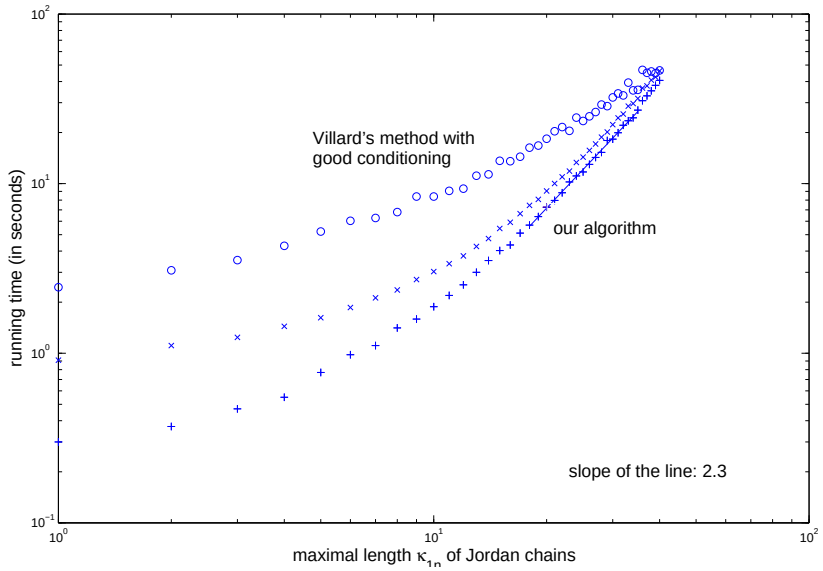

Fig. 6. Running time vs. κ_{1n}, the maximal Jordan chain length.

but the bulk of the time is spent computing the leading coefficients in the expansion of A, which is an artifact of using Matlab's symbolic toolbox to access Maple.

In the third test, we use 9×9 test matrices $D_k(\lambda)$ of the form

$$D_k(\lambda) = \mathrm{diag}[1, \ldots, 1, \lambda^k], \qquad (k = 1, 2, \ldots),$$

with $n = 9$, $l = 1$, $\kappa_{1n} = k$ and $d = k + 4$. The results are shown in Figure 6. As in the second test, Villard's method has an increasing cost as d grows. If we had access to the internal rref code, the re-use strategy for computing $\mathrm{rref}(\mathcal{A}_k)$ from $\mathrm{rref}(\mathcal{A}_{k-1})$ would decrease the running time of our algorithm by a factor of nearly κ_{1n}, i.e. the slope in Figure 6 would decrease by one.

As a final test, we use matrices $D_n(\lambda)$ similar to those in the first test, but with irreducible polynomials of higher degree. Specifically, we define

$$D_n(\lambda) = \mathrm{diag}[1, \ldots, 1, p_1, p_1 p_2, p_1^2 p_2, p_1^2 p_2^2],$$

where $p_1 = \lambda^2 + \lambda + 1$, $p_2 = \lambda^4 + \lambda^3 + \lambda^2 + 1$, $\kappa_{1n} = 2$, $\kappa_{2n} = 2$, $d = 16$ and n increases, starting at 4. The running times of the various algorithms are plotted in Figure 7.

4.1. Discussion

The key idea in our algorithm is that it is much less expensive to compute local Smith forms than global Smith forms through a sequence of elementary row and column operations. This is because (1) row reduction over R/pR in Algorithm 2 (or over K in the variant of Appendix A) is less expensive than computing extended GCDs over R; (2) the size of the rational numbers that occur in the algorithm remain smaller (as we only deal with the leading terms of A in an expansion in powers of p rather than with all of A); and (3) each column of $V(\lambda)$ in a local Smith form only has to be processed once for each power of p in the corresponding diagonal entry of $D(\lambda)$. Once the local Smith forms are known, we combine them to form a (global) multiplier $V(\lambda)$ for $A(\lambda)$. This

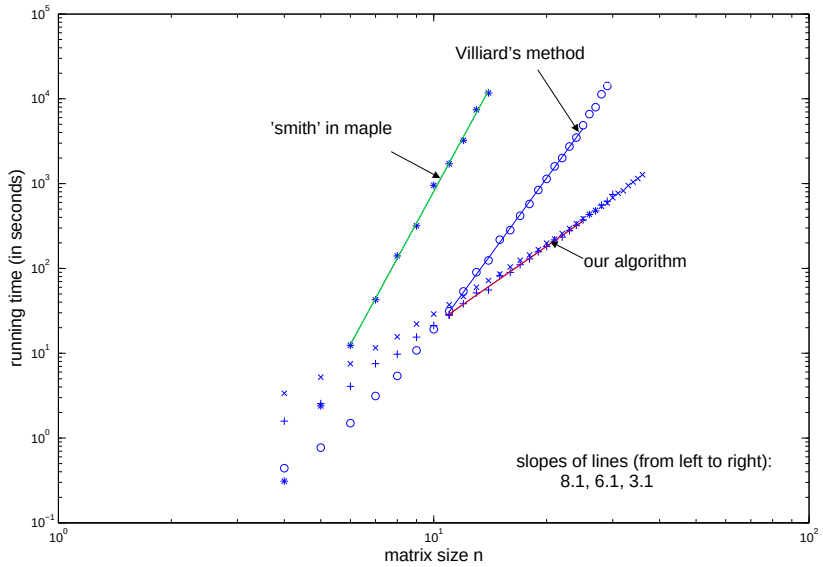

Fig. 7. Running time vs. matrix size, $\deg p_j(\lambda) > 1$.

last step does involve triangularization of $B_n(\lambda)$ via the extended GCD algorithm, but this is less time consuming in most cases than performing elementary row and column operations on $A(\lambda)$ to obtain $D(\lambda)$. This is due to the fact that we only have to apply row operations to $B_n(\lambda)$ (as the columns are already correctly ordered), and because the leading columns of $B_n(\lambda)$ tend to be sparse (as they consist of a superposition of local Smith forms, whose initial columns X_{-1} are a subset of the columns of the identity matrix). Sparsity is not used explicitly in our code, but it does reduce the work required to compute the extended GCD of a column.

The obvious drawback of our algorithm is that we have to compute a local Smith form for each irreducible factor of $\Delta(\lambda)$ separately, while much of the work in deciding whether to accept a column in Algorithm 1 can be done for all the irreducible factors simultaneously by using extended GCDs. In our numerical experiments, it appears that in most cases, the benefit of computing local Smith forms outweighs the fact that there are several of them to compute.

A. Alternative Version of Algorithm 2

In this section we present an algebraic framework for local Smith forms of matrix polynomials that shows the connection between Algorithm 2 and the construction of canonical systems of Jordan chains presented in [16]. This leads to a variant of the algorithm in which row-reduction is done in the field K rather than in R/pR.

Suppose R is a principal ideal domain and p is a prime in R. M defined via $M = R^n$ is a free R-module with a free basis $\{(1,0,\ldots,0),\ldots,(0,\ldots,1)\}$. Suppose $A : M \to M$ is a R-module morphism. We define submodules

$$N_k = \left\{x \in M : Ax \text{ is divisible by } p^{k+1}\right\}, \qquad (k \geq -1). \tag{A.1}$$

Then N_k is a free submodule of M by the structure theorem [5] for finitely generated modules over a principal ideal domain. (The structure theorem states that if M is a free module over a principal ideal domain R, then every submodule of M is free.) The rank of N_k is also n, as $p^{k+1}M \subset N_k \subset M$. Note that $N_{-1} = M$ and

$$N_k \subset N_{k-1}, \qquad (k \geq 0), \tag{A.2}$$

$$N_k \cap pM = pN_{k-1}, \qquad (k \geq 0). \tag{A.3}$$

Next we define the spaces W_k via

$$W_k = N_k/pN_{k-1}, \qquad (k \geq -1), \tag{A.4}$$

where $N_{-2} := M$ so that $W_{-1} = M/pM$. By (A.3), the action of R/pR on W_k is well-defined, i.e. W_k is a vector space over this field. Let us denote the canonical projection $M \to M/pM$ by π. Note that $\pi(pN_{k-1}) = 0$, so π is well-defined from W_k to M/pM for $k \geq -1$. It is also injective as $xp \in N_k \Rightarrow x \in N_{k-1}$, by (A.3). Thus, cosets $\{\dot{x}_1, \ldots, \dot{x}_m\}$ are linearly independent in W_k iff $\{\pi(x_1), \ldots, \pi(x_m)\}$ are linearly independent in M/pM. We define the integers

$$r_k = \text{dimension of } W_k \text{ over } R/pR, \qquad (k \geq -1) \tag{A.5}$$

and note that $r_{-1} = n$. We also observe that the truncation operator

$$id : W_{k+1} \to W_k : (x + pN_k) \mapsto (x + pN_{k-1}), \qquad (k \geq -1) \tag{A.6}$$

is well-defined ($pN_k \subset pN_{k-1}$) and injective ($x \in N_{k+1}$ and $x \in pN_{k-1} \Rightarrow x \in pN_k$, due to (A.3)). We may therefore consider W_{k+1} to be a subspace of W_k for $k \geq -1$, and have the inequalities

$$r_{k+1} \leq r_k, \qquad (k \geq -1). \tag{A.7}$$

The case $r_0 = 0$ is not interesting (as $N_k = p^{k+1}M$ for $k \geq -1$), so we assume that $r_0 > 0$. When $R = K[\lambda]$, which we assume from now on, this is equivalent to assuming that $\det[A(\lambda)]$ is divisible by $p(\lambda)$. We also assume that r_k eventually decreases to zero, say

$$r_k = 0 \quad \Leftrightarrow \quad k \geq \beta, \qquad \beta := \text{maximal Jordan chain length}. \tag{A.8}$$

This is equivalent to assuming $\det[A(\lambda)]$ is not identically zero.

The *while* loop of the algorithm in Algorithm 1 is a systematic procedure for computing a basis

$$\{x_j + pN_{k-2}\}_{j=n-r_{k-1}+1}^{n-r_k} \qquad (k \geq 0) \tag{A.9}$$

for a complement $\widetilde{W}_{k-1}$ of W_k in W_{k-1}:

$$\widetilde{W}_{k-1} \oplus W_k = W_{k-1}. \tag{A.10}$$

Any basis for any complement $\widetilde{W}_{k-1}$ would also lead to a local Smith form; however, the one we chose is particularly easy to compute and has the added benefit of yielding a unimodular multiplier V. The interpretation of the r_k as dimensions of the spaces W_k shows that the integers

$$\alpha_j = k, \qquad (n - r_{k-1} < j \leq n - r_k) \tag{A.11}$$

in the local Smith form are independent of the choices of complements $\widetilde{W}_{k-1}$ and bases (A.9) for $\widetilde{W}_{k-1}$. Using induction on k, it is easy to show that for any such choices, the vectors

$$\{(Ax_j/p^{\alpha_j}) + pM\}_{j=1}^{n-r_k} \tag{A.12}$$

are linearly independent in M/pM; otherwise, a linear combination of the form $\star$ in Algorithm 1 can be found which belongs to $\widetilde{W}_{k-1} \cap W_k$, a contradiction.

We now wish to find a convenient representation for these spaces suitable for computation. Since $p^{k+1}M \subset pN_{k-1}$, we have the isomorphism

$$N_k/pN_{k-1} \cong (N_k/p^{k+1}M)/(pN_{k-1}/p^{k+1}M), \tag{A.13}$$

i.e.

$$W_k \cong \mathbb{W}_k/p\mathbb{W}_{k-1}, \quad (k \geq 0), \qquad \mathbb{W}_k := N_k/p^{k+1}M, \quad (k \geq -1). \tag{A.14}$$

Although the quotient $\mathbb{W}_k/p\mathbb{W}_{k-1}$ is a vector space over R/pR, the spaces $\mathbb{W}_k$ and $M/p^{k+1}M$ are only modules over $R/p^{k+1}R$. They are, however, vector spaces over K, which is useful for developing a variant of Algorithm 2 in which row reduction is done over K instead of R/pR. Treating $M/p^{k+1}M$ as a vector space over K, $A(\lambda)$ induces a linear operator $\mathbb{A}_k$ on $M/p^{k+1}M$ with kernel

$$\mathbb{W}_k = \ker \mathbb{A}_k, \qquad (k \geq -1). \tag{A.15}$$

We also define

$$R_k = \frac{\text{dimension of } \mathbb{W}_k \text{ over } K}{s}, \qquad (k \geq -1, \quad s = \deg p) \tag{A.16}$$

so that $R_{-1} = 0$ and

$$R_k = r_0 + \cdots + r_k, \qquad (k \geq 0), \tag{A.17}$$

where we used $\mathbb{W}_0 = W_0$ together with (A.14) and the fact that as a vector space over K, $\dim W_k = sr_k$. By (A.11), $r_{k-1} - r_k = \#\{j \,:\, \alpha_j = k\}$, so

$$\begin{aligned} R_{\beta-1} &= r_0 + \cdots + r_{s-1} = (r_{-1} - r_0)0 + (r_0 - r_1)1 + \cdots + (r_{\beta-1} - r_\beta)\beta \\ &= \alpha_1 + \cdots + \alpha_n = \mu = \text{algebraic multiplicity of } p, \end{aligned} \tag{A.18}$$

where we used Theorem 7 in the last step. We also note that $\nu := r_0 = R_0 = s^{-1} \dim \ker(\mathbb{A}_0)$ can be interpreted as the geometric multiplicity of p.

Equations (A.14) and (A.15) reduce the problem of computing Jordan chains to that of finding kernels of the linear operators $\mathbb{A}_k$ over K. If we use the vector space isomorphism $\Lambda : K^{sn(k+1)} \to M/p^{k+1}M$ given by

$$\Lambda(x^{(0)}; \ldots; x^{(k)}) = \gamma(x^{(0)}) + p\gamma(x^{(1)}) + \cdots + p^k\gamma(x^{(k)}) + p^{k+1}M,$$

$$\gamma(x^{(j,1,0)}; x^{(j,1,1)}; \ldots; x^{(j,n,s-1)}) = \begin{pmatrix} x^{(j,1,0)} + \lambda x^{(j,1,1)} + \cdots + \lambda^{s-1}x^{(j,1,s-1)} \\ \cdots \\ x^{(j,n,0)} + \lambda x^{(j,n,1)} + \cdots + \lambda^{s-1}x^{(j,n,s-1)} \end{pmatrix}$$

to represent elements of $M/p^{k+1}M$, then multiplication by λ in $M/p^{k+1}M$ viewed as a module becomes the following linear operator on $K^{sn(k+1)}$ viewed as a vector space over K:

$$\mathbb{S}_k = \begin{pmatrix} I \otimes S & 0 & 0 & 0 \\ I \otimes Z & I \otimes S & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & I \otimes Z & I \otimes S \end{pmatrix}, \quad S \text{ as in (34)}, \quad Z = \begin{pmatrix} 0 & 0 & 1 \\ & \ddots & 0 \\ 0 & & 0 \end{pmatrix}. \tag{A.19}$$

Here $\mathbb{S}_k$ is a $(k+1)\times(k+1)$ block matrix, $I\otimes S$ is a Kronecker product of matrices, S and Z are $s\times s$ matrices, and I is $n\times n$. Multiplication by λ^m is represented by $\mathbb{S}_k^m$, which has a similar block structure to $\mathbb{S}_k$, but with S replaced by S^m and Z replaced by

$$T_m = \begin{cases} 0 & m = 0 \\ \sum_{l=0}^{m-1} S^l Z S^{m-1-l}, & 1 \le m \le s-1. \end{cases} \tag{A.20}$$

Thus, if we expand

$$A(\lambda) = A^{(0)} + pA^{(1)} + \cdots + p^q A^{(q)}, \qquad A^{(j)} = A^{(j,0)} + \cdots + \lambda^{s-1} A^{(j,s-1)}, \tag{A.21}$$

the matrix $\mathbb{A}_k$ representing $A(\lambda)$ is given by

$$\mathbb{A}_k = \begin{pmatrix} A_0 & 0 & \cdots & 0 \\ A_1 & A_0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ A_k & A_{k-1} & \cdots & A_0 \end{pmatrix}, \tag{A.22}$$

where

$$A_j = \begin{cases} \sum_{m=0}^{s-1} A^{(0,m)} \otimes S^m, & r = 0, \\ \sum_{m=0}^{s-1} \left[A^{(j,m)} \otimes S^m + A^{(j-1,m)} \otimes T_m \right], & r \ge 1. \end{cases} \tag{A.23}$$

The terms $\sum_m A^{(j,m)} \otimes S^m$ and $\sum_m A^{(j-1,m)} \otimes T_m$ compute the remainder and quotient in (28) if we set $y = \mathbb{A}_k \cdot (x^{(0)}; \ldots; x^{(k)})$ and compare $y^{(k)}$ to the formula for $y_i = (Ax_i/p^k) \,\%\, p$ in (28).

Next we seek an efficient method of computing a basis matrix $\mathbb{X}_k$ for the nullspace $\mathbb{W}_k = \ker \mathbb{A}_k$. Note that we are now working over the field K rather than R/pR as we did in Section 3. Suppose $k \ge 1$ and we have computed $\mathbb{X}_{k-1}$. The first k blocks of equations in $\mathbb{A}_k\mathbb{X}_k = 0$ imply there are matrices $\mathbb{U}_k$ and $\mathbb{V}_k$ such that $\mathbb{X}_k = [\mathbb{X}_{k-1}\mathbb{U}_k; \mathbb{V}_k]$, while the last block of equations is

$$\overbrace{\begin{pmatrix} A_k & \ldots & A_0 \end{pmatrix} \underbrace{\begin{pmatrix} 0 & \mathbb{X}_{k-1} \\ I_{sn\times sn} & 0 \end{pmatrix}}_{\mathbb{X}_k} \begin{pmatrix} \mathbb{V}_k \\ \mathbb{U}_k \end{pmatrix}}^{\mathcal{A}_k} = \begin{pmatrix} 0_{sn\times sR_k} \end{pmatrix}. \tag{A.24}$$

Thus, we can build up the matrices $\mathbb{X}_k$ recursively via $\mathbb{X}_0 = \mathrm{null}(A_0)$ and

$$\mathcal{A}_k = \begin{pmatrix} A_0 \,,\, [A_k, \ldots, A_1]\mathbb{X}_{k-1} \end{pmatrix}, \qquad [\mathbb{V}_k; \mathbb{U}_k] = \mathrm{null}(\mathcal{A}_k), \qquad \mathbb{X}_k = [\mathbb{X}_{k-1}\mathbb{U}_k; \mathbb{V}_k].$$

Here $\mathrm{null}(\cdot)$ is the standard algorithm for computing a basis for the nullspace of a matrix by reducing it to row-echelon. As the first sn columns of $\mathcal{A}_1$ coincide with $\mathcal{A}_0 := A_0$, and since $\mathbb{X}_0 = \mathrm{null}(\mathcal{A}_0)$, there are matrices V_1 and U_1 such that

$$\begin{pmatrix} \mathbb{V}_1 \\ \mathbb{U}_1 \end{pmatrix} = \begin{pmatrix} \mathbb{X}_0 & V_1 \\ 0 & U_1 \end{pmatrix}, \qquad \mathbb{X}_1 = \left[\iota(\mathbb{X}_0), X_1\right], \qquad X_1 = [\mathbb{X}_0 U_1; V_1], \tag{A.25}$$

where $\iota : K^{snl} \to K^{sn(l+1)}$ represents multiplication by p from M/p^lM to $M/p^{l+1}M$:

$$\iota(x^{(0)}; \ldots; x^{(l-1)}) = (0; x^{(0)}; \ldots; x^{(l-1)}), \qquad x^{(j)},\, 0 \in K^{sn}. \tag{A.26}$$

But now the first $s(n+R_0)$ columns of $\mathcal{A}_2$ coincide with $\mathcal{A}_1$, so there are matrices V_2 and U_2 such that

$$\begin{pmatrix}\mathbb{V}_2\\ \mathbb{U}_2\end{pmatrix} = \begin{pmatrix}\mathbb{V}_1 & V_2\\ [\mathbb{U}_1;0] & U_2\end{pmatrix}, \qquad \mathbb{X}_2 = \big[\iota(\mathbb{X}_1), X_2\big], \qquad X_2 = [\mathbb{X}_1 U_2; V_2]. \tag{A.27}$$

Continuing in this fashion, we find that the first $s(n+R_{k-2})$ columns of $\mathcal{A}_k$ coincide with $\mathcal{A}_{k-1}$, and therefore $\mathbb{V}_k$, $\mathbb{U}_k$ and $\mathbb{X}_k$ have the form

$$\begin{pmatrix}\mathbb{V}_k\\ \mathbb{U}_k\end{pmatrix} = \begin{pmatrix}X_0 & V_1 & \cdots & V_{k-1} & V_k\\ 0 & [U_1;0] & \cdots & [U_{k-1};0] & U_k\end{pmatrix}, \qquad X_0 = \mathbb{X}_0, \tag{A.28}$$

$$\mathbb{X}_k = \left[\begin{pmatrix}0_{snk\times sr_0}\\ X_0\end{pmatrix}, \begin{pmatrix}0_{sn(k-1)\times sr_1}\\ X_1\end{pmatrix}, \ldots, \begin{pmatrix}X_k\end{pmatrix}\right], \qquad X_k = [\mathbb{X}_{k-1}U_k; V_k].$$

By construction, $\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k]$ is a basis for $\mathbb{W}_k$ when $k \geq 1$; it follows that $X_k + \iota(\mathbb{W}_{k-1})$ is a basis for W_k when W_k is viewed as a vector space over K. We define $X_0 = \mathbb{X}_0$ and $X_{-1} = I_{sn\times sn}$ to obtain bases for W_0 and W_{-1} as well.

But we actually want a basis for W_k viewed as a vector space over R/pR rather than K. Fortunately, all the matrices in this construction are manipulated in $s\times s$ blocks; everywhere we have an entry in R/pR in the construction of Section 3, we now have an $s\times s$ block with entries in K. This is because the matrix $\mathbb{A}_k$ commutes with $\mathbb{S}_k$ (since $A(\lambda)$ commutes with multiplication by λ). So if we find a vector $x \in \ker \mathbb{A}_k$, the vectors

$$\{x\,,\, \mathbb{S}_k x\,,\, \ldots\,,\, \mathbb{S}_k^{s-1}x\} \tag{A.29}$$

will also belong to $\ker \mathbb{A}_k$. These vectors are guaranteed to be linearly independent, for otherwise the vectors

$$\{x^{(j,i)}\,,\, Sx^{(j,i)}\,,\, \ldots\,,\, S^{s-1}x^{(j,i)}\} \tag{A.30}$$

would be linearly dependent in K^s, where $x^{(j,i,m)}$ is the first non-zero entry of x. This is impossible since p is irreducible; see (33) above. As a consequence, in the row-reduction processes, if we partition $\mathcal{A}_k$ into groups of s columns, either all s columns will be linearly dependent on their predecessors, or each of them will start a new row in $\mathrm{rref}(\mathcal{A}_k)$; together they contribute a block identity matrix $I_{s\times s}$ to $\mathrm{rref}(\mathcal{A}_k)$. It follows that the block nullspace matrices $[\mathbb{V}_k;\mathbb{U}_k]$ will have supercolumns (groups of s columns) that terminate in identity matrices $I_{s\times s}$, and that the submatrix $X_k^{(0)}$ consisting of the first ns rows of X_k also has supercolumns that terminate with $I_{s\times s}$. The structure is entirely analogous to the one in Figure 2 above, with $s\times s$ blocks replacing the entries of the matrices shown. The following four conditions then ensure that successive columns in a supercolumn of X_k are related to each other via (A.29): (1) the terminal identity blocks in $X_k^{(0)}$ contain the only non-zero entries of their respective rows; (2) if x is a column of X_k and $x^{(0,i)} \in K^s$ is one of the columns of a terminal identity block, the $(\mathbb{S}_k x)^{(0,i)} = Sx^{(0,i)}$ is the next column of $I_{s\times s}$ (with the 1 shifted down a slot). (3) the columns of $\mathbb{X}_k$ are a basis for $\ker \mathbb{A}_k$; (4) if x is a column of $\mathbb{X}_k$, then $\mathbb{S}_k x$ also belongs to $\ker \mathbb{A}_k$. Thus, to extract a basis for W_k over R/pR, we simply extract the first column of each supercolumn of X_k. The result is identical to that of Algorithm 2. In practice, this version of the algorithm (working over K) is easier to implement, but the other version (over R/pR) should be about s times faster as the cost of multiplying two elements of R/pR is $O(s^2)$ while the

cost of multiplying two $s \times s$ matrices is $O(s^3)$. The results in Figure 7 were computed as described in this appendix (over $K = \mathbb{Q}$).

References

[1] F.R. Gantmacher. *Matrix Theory*, volume 1. Chelsea Publishing Company, 1960.

[2] I. Gohberg, M. A. Kaashoek, and F. van Schagen. On the local theory of regular analytic matrix functions. *Linear Algebra Appl.*, 182:9–25, 1993.

[3] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix Polynomials*. Academic Press, New York, 1982.

[4] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, 1996.

[5] Thomas W. Hungerford. *Algebra*. Springer, New York, 1996.

[6] E. Kaltonfen, M.S. Krishnamoorthy, and B.D. Saunders. Fast parallel computation of Hermite and Smith forms of polynomial matrices. *SIAM J. Alg. Disc. Meth.*, 8(4):683–690, 1987.

[7] E. Kaltonfen, M.S. Krishnamoorthy, and B.D. Saunders. Parallel algorithms for matrix normal forms. *Linear Algebra and its Applications*, 136:189–208, 1990.

[8] R. Kannan. Solving systems of linear equations over polynomials. *Theoretical Computer Science*, 39:69–88, 1985.

[9] S. Labhalla, H. Lombardi, and R. Marlin. Algorithmes de calcul de la rduction de Hermite d'une matrice coefficients polynomiaux. *Theoretical Computer Science*, 161:69–92, 1996.

[10] A. Storjohann. Computation of Hermite and Smith normal forms of matrices. 1994.

[11] A. Storjohann and G. Labahn. A fast Las Vegas algorithm for computing the Smith normal form of a polynomial matrix. *Linear Algebra and its Applications*, 253:155–173, 1997.

[12] G. Villard. Computation of the Smith normal form of polynomial matrices. In *International Symposium on Symbolic and Algebraic Computation, Kiev, Ukraine*, pages 209–217. ACM Press, 1993.

[13] G. Villard. Fast parallel computation of the Smith normal form of polynomial matrices. In *International Symposium on Symbolic and Algebraic Computation, Oxford, UK*, pages 312–317. ACM Press, 1994.

[14] G. Villard. Generalized subresultants for computing the Smith normal form of polynomial patrices. *J. Symbolic Computation*, 20:269–286, 1995.

[15] G. Villard. Fast parallel algorithms for matrix reduction to normal forms. *Appli. Alg. Eng. Comm. Comp.*, 8(6):511–538, 1997.

[16] J. Wilkening. An algorithm for computing Jordan chains and inverting analytic matrix functions. *Linear Algebra Appl.*, 427/1:6–25, 2007.